Apple IIe
#6: The Apple II Paddle Circuits

Revised by: Glenn A. Baxter November 1988
Written by: Peter Baum May 1984

This Technical Note describes the paddle circuit used in the Apple II family of computers.

Caveats

Since Apple has introduced machines with internal clock speeds which may not be exactly 1.023 MHz, it is best to use the `PREAD` firmware call to read paddle data. This Note assumes that the clock speed of the system is exactly 1.023 MHz. If you want to insure accuracy in reading paddle data, you should make sure the system is first running at the correct speed. Enough information is provided so that you can write your own `PREAD` routine, although this is discouraged. If the program runs on an Apple IIGS or some future machine, your custom paddle reading routine will fail to give the correct results.

Circuit Description

The value of the Apple paddles (or joystick) is determined by a software timing loop reading a change of state in a timing circuit. The paddles consist of a variable resistor (from 0-150k ohms) which makes up part of the timing circuit. There is a routine in the monitor ROM, called `PREAD`, which counts the time until a state change occurs in the paddle circuit. This time is translated into a value between 0 and 255.

The block diagrams in Figures 1 and 2 show the paddle circuit for the Apple][+, Apple IIc, and the Apple IIe. The large block on the left illustrates part of the circuitry inside the 558 timer chip. The 558 chip consists of four of these blocks, with all four paddle triggers lines shorted together on the motherboard and activated by the soft switch at $C070. The outputs of the 558 chip run into a multiplexer, which places the appropriate signal onto the high bit of the data bus when a paddle soft switch address in the range $C064 $C067 is read. The Apple IIc uses a 556 timer rather than the 558 chip and only supports two paddles, 0 and 1.

The 100 ohm resistor and .022 microfarad capacitor are on the motherboard, with the variable resistor in the paddle. Each of the four paddle inputs have their own capacitor and resistor. Since these components can vary by as much as five percent from Apple to Apple, this circuit is not a very exact analog to digital converter. If a paddle is moved from one Apple to another

without changing the resistance (turning the knob), the paddle read routine will probably calculate a different value for each machine. About the only feature of the paddle read routine that a programmer can depend on is that the value returned will rise if the paddle resistance increases (or fall if the resistance decreases).

The paddle timing circuit on the Apple][+ and Apple IIc is slightly different than the one on the Apple IIe. On the Apple IIe, the 100 ohm fixed resistor is between the transistor and the capacitor, while the variable resistor in the paddle is connected directly to the capacitor. On the Apple][+ and IIc, the capacitor is connected directly to the transistor and the fixed resistor is in series with paddle resistor.

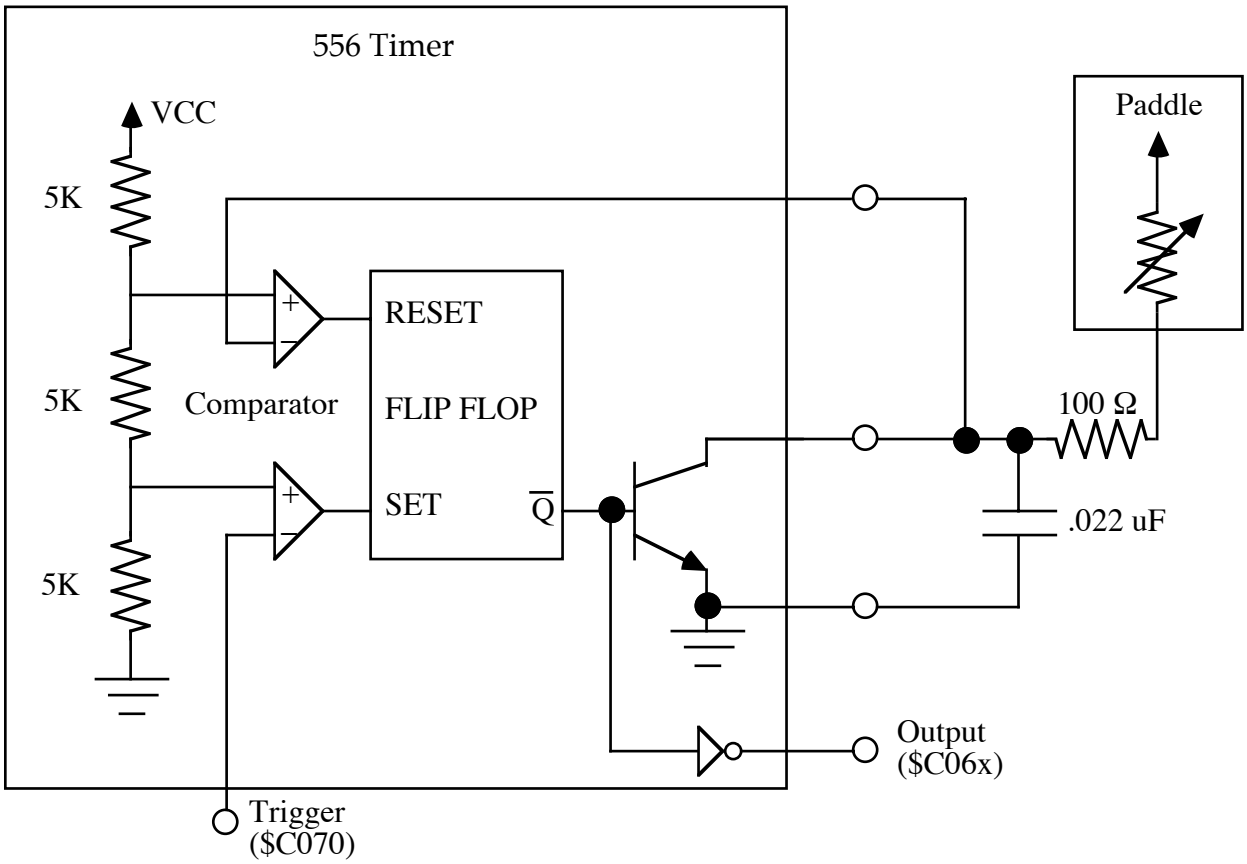

Figure 1–Apple][+ and IIc Paddle Circuit

An Example of a Typical Paddle Read Routine

The timing circuit works by discharging a capacitor through a transistor, then shutting the transistor off and letting the paddle charge the capacitor by supplying current through the variable resistor. The rate at which the capacitor charges is a function of the variable resistance; the lower the paddle resistance, the greater the current and the faster the capacitor charges. When the capacitor reaches a predetermined value it changes the state of a flip flop. The paddle read routine counts the time it takes for the capacitor to rise and change the flip flop.

Let's step through an example of a typical paddle read operation. For now we will assume the capacitor has already been discharged and in a few pages I will explain when this assumption can be made and when it cannot.

The software starts by reading the soft switch at location $C070, which strobes the trigger lines on the 558 timer. This action causes two events to occur, the output signal (which is read at $C064-$C067 for paddle 0-3, respectively) goes high and the transistor turns off.

The software, after initially strobing the trigger line, executes a timing loop which reads the state of the output signal. When the output signal changes from high to low the software jumps out of the timing loop and returns a value indicating the time. The monitor `PREAD` routine consists of a 11 μsec. loop and will return a value between 0 and 255. (Note: The firmware listing is wrong and says the loop is 12 μsec.) The timing loop returns 255 if the circuit takes longer than 2.82 ms for the state change to occur.

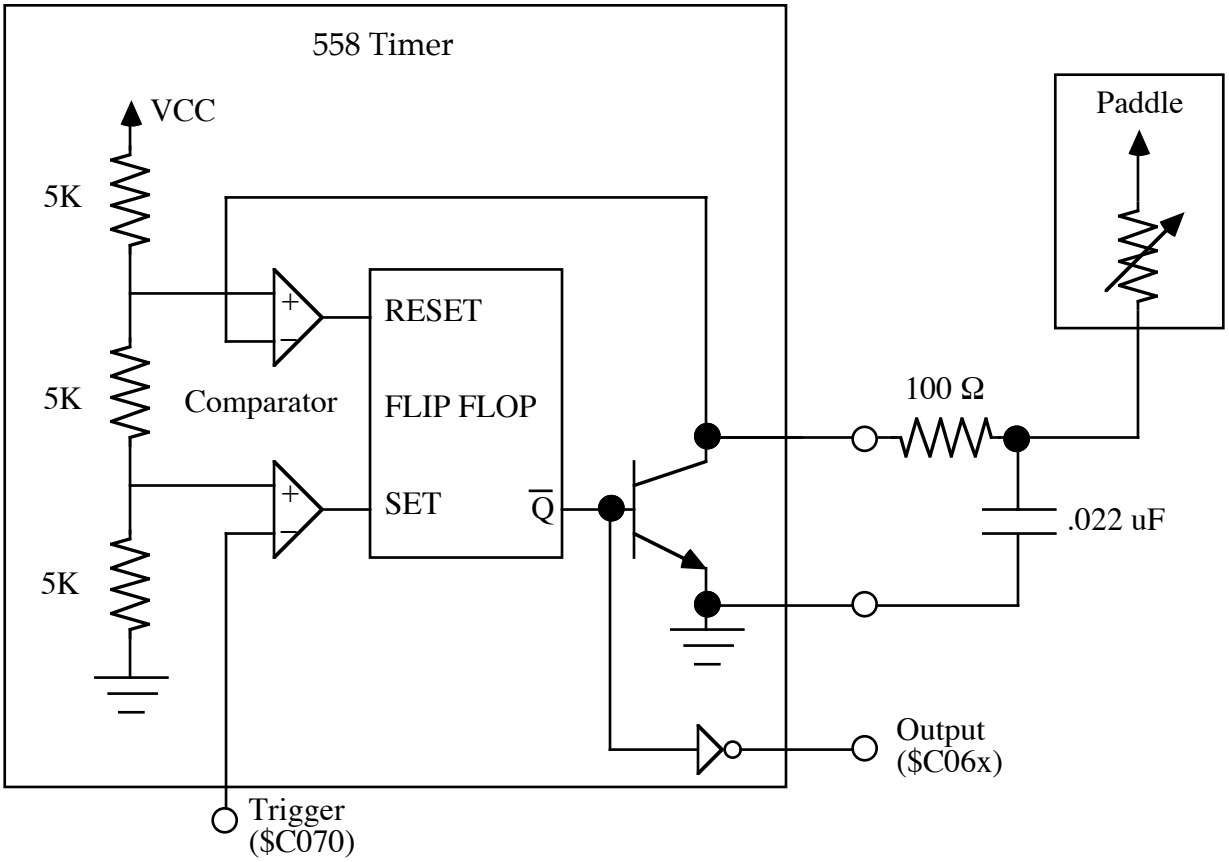

Figure 2–Apple IIe Paddle Circuit

```
* PADDLE READ ROUTINE
* ENTER WITH PADDLE NUMBER (0-3) IN X-REG

FB1E:AD 70 C0  PREAD   4  LDA  PTRIG     ;TRIGGER PADDLES
FB21:A0 00             2  LDY  #0        ;INIT COUNTER
FB23:EA                2  NOP            ;COMPENSATE FOR 1ST COUNT
FB24:EA                2  NOP
FB25:BD 64 C0  PREAD2  4  LDA  PADDL0,X  ;COUNT EVERY 11 µSEC.
```

```
FB28:10 04              2  BPL  RTS2D     ;BRANCH WHEN TIMED OUT
FB2A:C8                 2  INY            ;INCREMENT COUNTER
FB2B:D0 F8              3  BNE  PREAD2    ;CONTINUE COUNTING
FB2D:88                    DEY            ;COUNTER OVERFLOWED
FB2E:60         RTS2D      RTS            ;RETURN W/VALUE 0-255
```

Inside the 558 timer chip, when the trigger is strobed low, the comparator that feeds the set input of the flip flop is triggered, which in turn sets the output of the 558 timer. At the same time, the transistor, which has held the capacitor near ground by sinking current from it, is shut off. The capacitor can now charge using the current supplied by the paddle. The smaller the paddle's resistance, the more current the paddle will supply and the faster the capacitor charges. After some time, the capacitor will charge to the threshold value of 3.3 volts, which is set by the voltage divider network in the 558 timer, and the comparator that feeds the reset input on the flip flop will trigger. This trigger sets the output signal ($C06x) of the 558 timer low, which indicates to the software that the circuit has timed out.

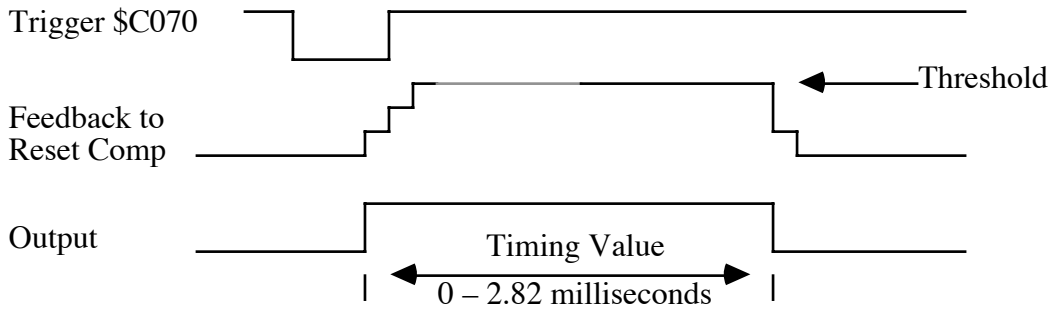

Figure 3–Paddle Circuit Recharge Timing

Resetting the flip flop turns the transistor on, which discharges the capacitor very quickly (normally less than 250 ns). That paddle can then be read again.

A Closer Look at the Hardware

The First Anomaly

Notice that the last sentence states that the paddle can be read again and not the paddles. If another paddle is read immediately after the first, it may yield the wrong value. To demonstrate this, I will step through an example of reading a second paddle immediately after finishing the first.

In this example I will assume that the first paddle has been set with a very low resistance, while the second paddle has a high resistance. The first paddle will time out very quickly and return with a small value, while the second paddle will take longer and yield a larger value.

We start reading the paddles by testing the paddle outputs to see if they are low, which indicates that the capacitor has been discharged. Assuming that the outputs are low, the next step is to trigger the 558 timer ($C070), which turns off the transistor and allows the capacitors to charge. Since all of the trigger input lines are shorted together, all four of the capacitors will charge, but

at different rates since the paddle resistances have been set to different values. The voltage on the capacitor for the first paddle will reach the threshold voltage very quickly since the paddle resistance has been set low, therefore the timing loop will time out quickly.

At this point the capacitor for the second paddle is still charging and has not yet reached the threshold since the paddle resistance was set to a high value. The transistor for the second paddle is still turned off due to the initial trigger used for reading paddle one. This means that the capacitor for the second paddle has not been discharged.

Any attempts at reading the second paddle now will only yield false results. The capacitor is partly charged and therefore will reach the threshold value much faster than if the capacitor had been completely discharged. If the timing loop is used, it will return with a smaller value than it would if the capacitor had been completely discharged. Notice that retriggering (reading location $C070) the 558 timer will not help, since that only keeps the transistor turned off and does not help discharge the capacitor. The only way for the capacitor to discharge is to let the circuit time out completely by letting the capacitor charge until it resets the flip flop.

To read the second paddle, the capacitor must first be discharged, which is only done when the threshold value is reached and the 558 timer flip flop is reset. The only way to guarantee that the capacitor is discharged is if the transistor is on. This condition is met when the paddle output is low. Therefore, start every paddle read either by waiting for at least 3 ms before strobing the trigger input or testing to make sure that the paddle output is low.

If after 4 ms the paddle output is not low, then there is a good chance that there is no paddle connected. This result may also indicate that a peripheral with a larger maximum value resistor than the 150k ohms used by the Apple paddles is attached. Some peripheral devices use this technique of a larger variable resistor so that more than 256 points of resolution can be determined. Of course, this requires a custom software driver and the monitor `PREAD` routine cannot be used.

Apple IIe Anomalies

The problem with Apple IIe paddle input is that the capacitor may not be discharged by the transistor. Typically, the transistor will discharge the capacitor in less than 250 ns on the Apple][+. But on the Apple IIe, if the paddle resistance is very low then the paddle may supply enough current to always keep the capacitor charged.

Because the fixed resistor (100 ohms) on the Apple IIe motherboard is between the capacitor and the transistor, there will be a voltage drop across the resistor if the capacitor stays charged. When the transistor is shut off by the trigger strobe, this voltage drop will disappear and the capacitor, which may be near the threshold voltage, will trigger the reset comparator earlier than it would if the capacitor had been discharged completely. The net affect of this is that the paddles will read zero on the Apple IIe when they would read a small value on the Apple][+ or IIc.

Other circuits which expect the capacitor to discharge completely may not work properly. A circuit which attempts to simulate a paddle through active components such as a digital to analog

converter may be able to source enough current that the capacitor never discharges and the paddle always reads zero.

It should also be noted that due to electromagnetic interference, later model IIe computers actually have an extra capacitor attached between the `BUTTON` inputs and ground. This essentially **slows** the response time of the input, making a fully digital input appear a bit more analog (no pun intended). Care should be taken in designing system which depend on a certain repetition rate of the button inputs. Careful engineering and testing across systems should prevent any problems. As an example, adding a transistor output stage to drive the button inputs to the appropriate states might be a good idea for a serializing A/D. A joystick would not require this kind of circuit because the user input is too slow to be affected by the capacitors. For more information on the changes in later model IIe computers, refer to Apple IIe Technical Note #9, Switch Input Changes.

Conclusion

Hopefully, this Note has given the reader a good feel for the paddle circuitry and the routines which determine the paddle values. To reinforce the material covered, you should try writing your own paddle read routine. For example, you could write a read routine that would read two paddles at once. The software loop will not have the 11 μsec. resolution of the `PREAD` routine, but you will find it still works just fine. Happy programming.

Further Reference

- *Apple IIe Technical Reference Manual*